Log: Week 8

This week I have:

- Implemented basic/ placeholder genetic operators
- These allow the algorithm to accurately learn/ approximate straight line routes from $A \to B$

Below you can see the population after 1 generation:

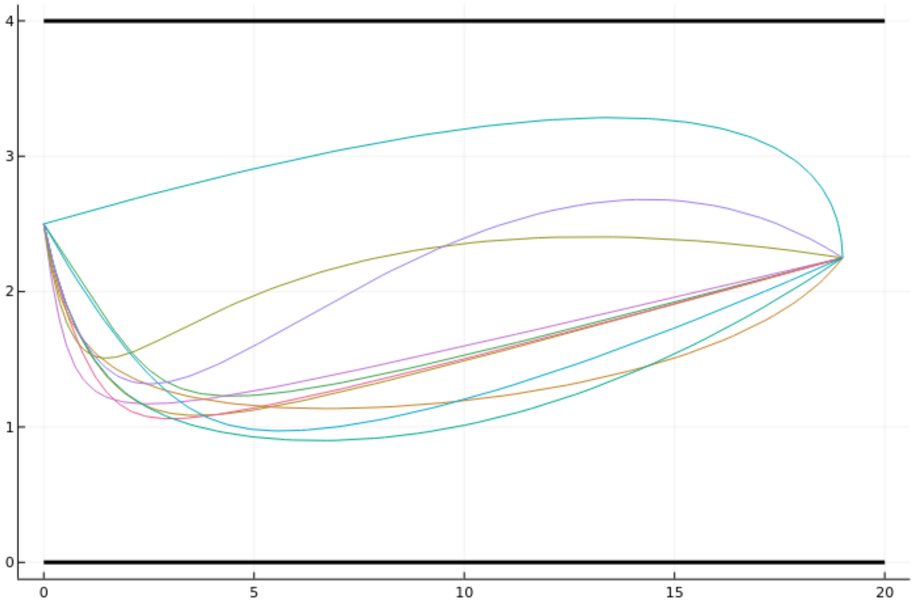

Figure 1: 1 Generation

And the most and least fit solutions after 4 generations

Whilst in isolation this is not particularly impressive, in fact is under performs a simple Pythagorean distance calculation, it can be generalised and extended to avoid obstacles and to not intersect with other routes.

I went on to implement road space obstacles in the form of circles. I have implemented the abstract type of `Obstacle` of which `Circle` is a member allowing me to extend my program.

By calculating infeasible route sections as the distance between 2 intersects between an individual and a obstacle, I can penalise solutions where such a distance is non-zero.

The results can be seen in Figure 3. These results are seen after just a single

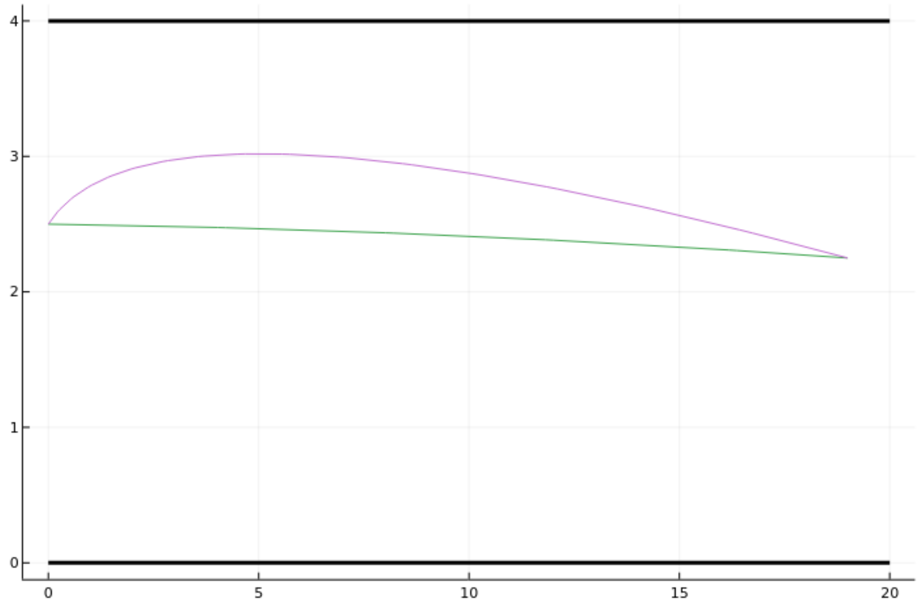

Figure 2: 4 Generations min max

generation with the best solution having approximately 8% lower fitness than the Pythagorean route.

I have now moved on to initial parameter tweaking in order to get the best performance from these basic operators. I have been testing different weightings for infeasible and high proximity routes, I need to make sure they are never the best solution but not penalise them too much so that solutions near them in the search space are never explored; the optimal solution could very well be close to an infeasible one.

You can find a gif of the algorithm over the course of 100 generations here. The bold line is the fittest and the control points for the best curve are shown. Obstacles are shown in red.

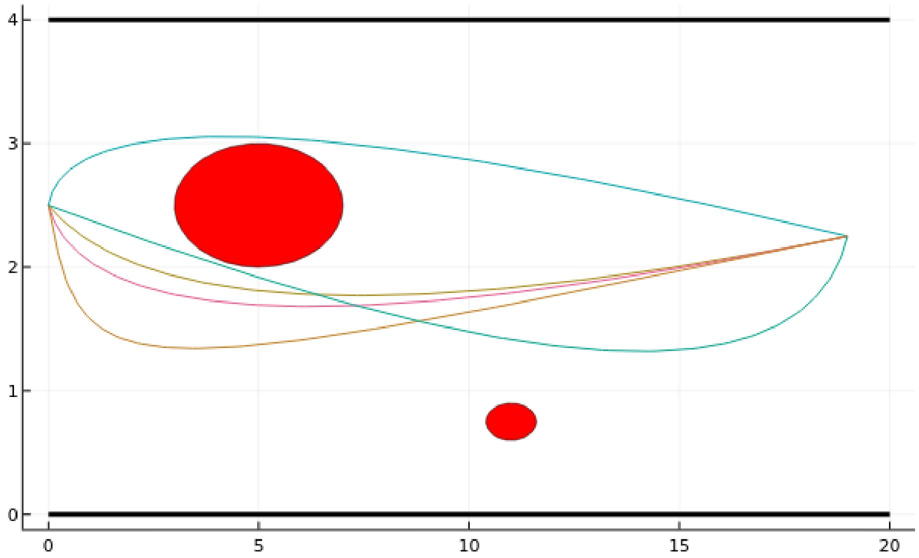

Figure 3: Obstacle avoidance